CE263: Database Management System

Formal Relational Query Languages

Devang Patel Institute of Advance Technology and Research

Outline

- Relational Algebra
- Tuple Relational Calculus
- Domain Relational Calculus

Relational Algebra

- Procedural language
- Six basic operators
  - select: σ
  - project: $\prod$
  - union: $\cup$
  - set difference: –
  - Cartesian product: x
  - rename: ρ
- The operators take one or two relations as inputs and produce a new relation as a result.

Relation used

- Instructor Relation
  - ID, Name, Dept_name, Salary
- Section Relation
  - Course_id, Sec_id, Semester, Year, Building, Room_no, Time_slot_id
- Teaches Relation
  - ID, Course_ID, Sec_id, Semester, Year

Select Operation

- Notation: $\sigma_{p}(r)$
- Defined as:

  $\sigma_p(\mathbf{r})$
  Where p is a formula in propositional calculus consisting of **terms** connected by : $\land$ (**and**), $\lor$ (**or**), $\neg$ (**not**)
  Each **term** is one of:

  <attribute> *op* <attribute> or <constant>

  where *op* is one of: $=, \neq, >, \geq. <. \leq$

- Example of selection:

  $\sigma_{dept\_name=\text{“}Physics\text{”}}(instructor)$

Project Operation

- Notation:

$$\Pi_{A_1, A_2, \ldots, A_k}(r)$$

  where A_1, A_2 are attribute names and r is a relation name.

- The result is defined as the relation of k columns obtained by erasing the columns that are not listed
- Duplicate rows removed from result, since relations are sets
- Example: To eliminate the *dept_name* attribute of *instructor*

$$\Pi_{ID,\ name,\ salary}(instructor)$$

Union Operation

- Notation: $r \cup s$
- Defined as:

  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$
- For $r \cup s$ to be valid.
  1. *r, s* must have the *same* **arity** (same number of attributes)
  2. The attribute domains must be **compatible** (example: 2nd column of *r* deals with the same type of values as does the 2nd column of *s*)
- Example: to find all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or in both

$$\Pi_{course\_id}(\sigma_{semester=\text{“Fall”} \land year=2009}(section)) \cup$$
$$\Pi_{course\_id}(\sigma_{semester=\text{“Spring”} \land year=2010}(section))$$

Set Difference Operation

- Notation $r - s$
- Defined as:

  $r - s = \{t \mid t \in r$ **and** $t \notin s\}$

- Set differences must be taken between **compatible** relations.
  - r and s must have the same arity
  - attribute domains of r and s must be compatible
- Example: to find all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\Pi_{course\_id} (\sigma_{semester=\text{“Fall”} \ \land \ year=2009} (section)) \ -$$
$$\Pi_{course\_id} (\sigma_{semester=\text{“Spring”} \ \land \ year=2010} (section))$$

Set-Intersection Operation

- Notation: $r \cap s$
- Defined as:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Assume:
  - *r*, *s* have the *same arity*
  - attributes of *r* and *s* are compatible
- Note: $r \cap s = r - (r - s)$

Cartesian-Product Operation

- Notation $r \times s$
- Defined as:

  $r \times s = \{t\,q \mid t \in r$ **and** $q \in s\}$

σ branch-name = “Perryridge”(borrower × loan)

Rename Operation

- Assume that a relational algebra expression E has arity n.
- If a relational-algebra expression *E* has arity *n*, then

$$\rho_{x(A_1, A_2, \ldots, A_n)}(E)$$

returns the result of expression *E* under the name *X*, and with the attributes renamed to $A_1, A_2, \ldots, A_n$.

Division Operation

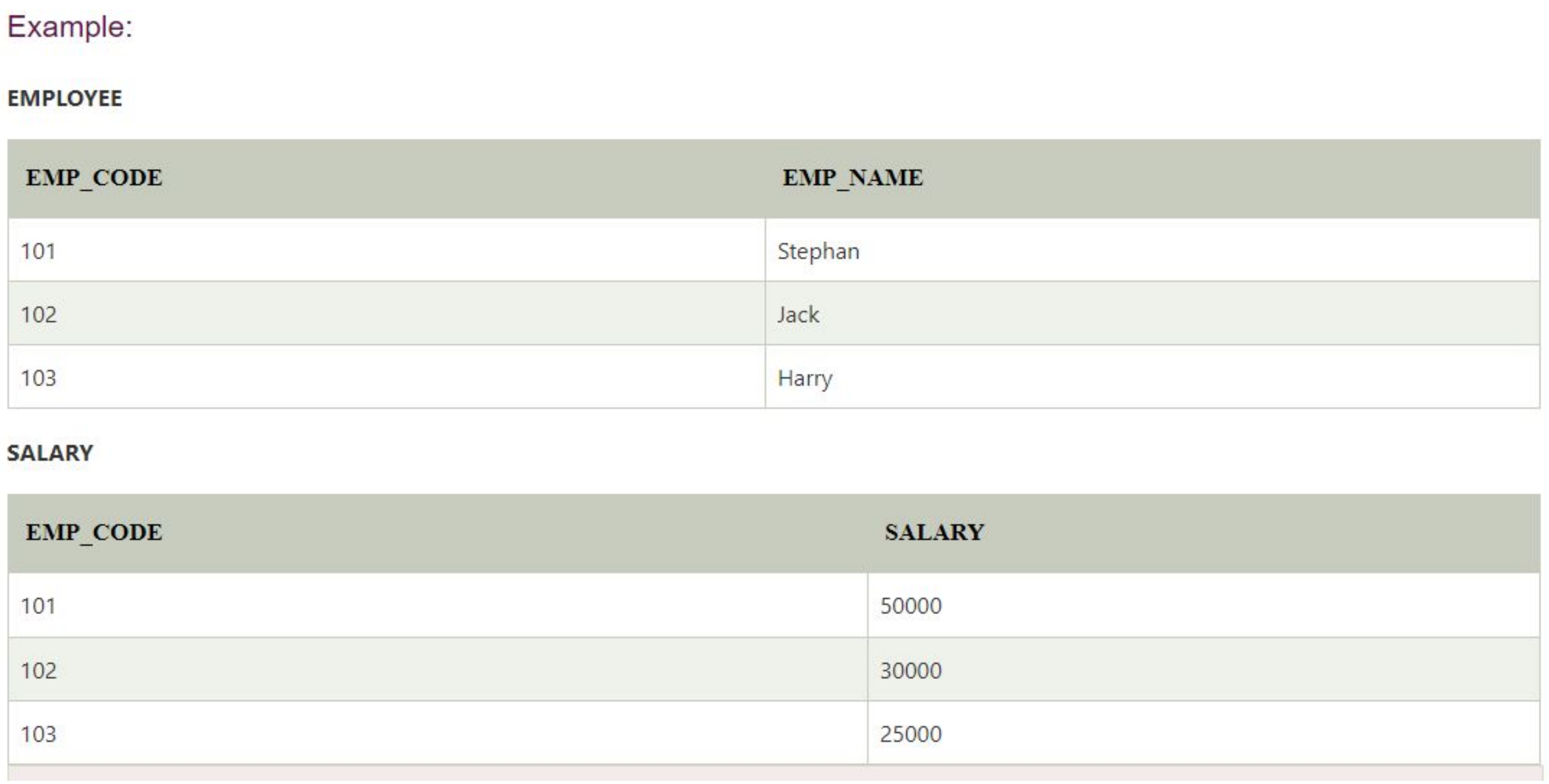
- Formally, let r(R) and s(S) be relations, and let S ⊆ R; that is, every attribute of schema S is also in schema R.
- The relation r ÷ s is a relation on schema R – S (that is, on the schema containing all attributes of schema R that are not in schema S).

Formal Definition

- A basic expression in the relational algebra consists of either one of the following:
  - A relation in the database
  - A constant relation
- Let E_1 and E_2 be relational-algebra expressions; the following are all relational-algebra expressions:
  - $E_1 \cup E_2$
  - $E_1 - E_2$
  - $E_1 \times E_2$
  - $\sigma_p (E_1)$, P is a predicate on attributes in E_1
  - $\prod_S(E_1)$, S is a list consisting of some of the attributes in E_1
  - $\rho_x (E_1)$, x is the new name for the result of E_1

Join Operation

- A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied.
- It is denoted by ⋈.



Example:

EMPLOYEE

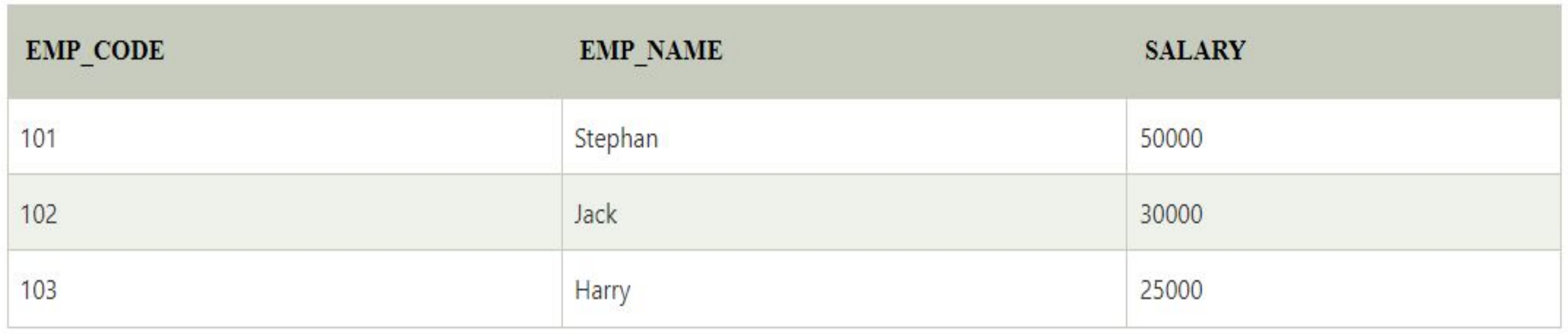
EMP_CODE	EMP_NAME
101	Stephan
102	Jack
103	Harry

SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

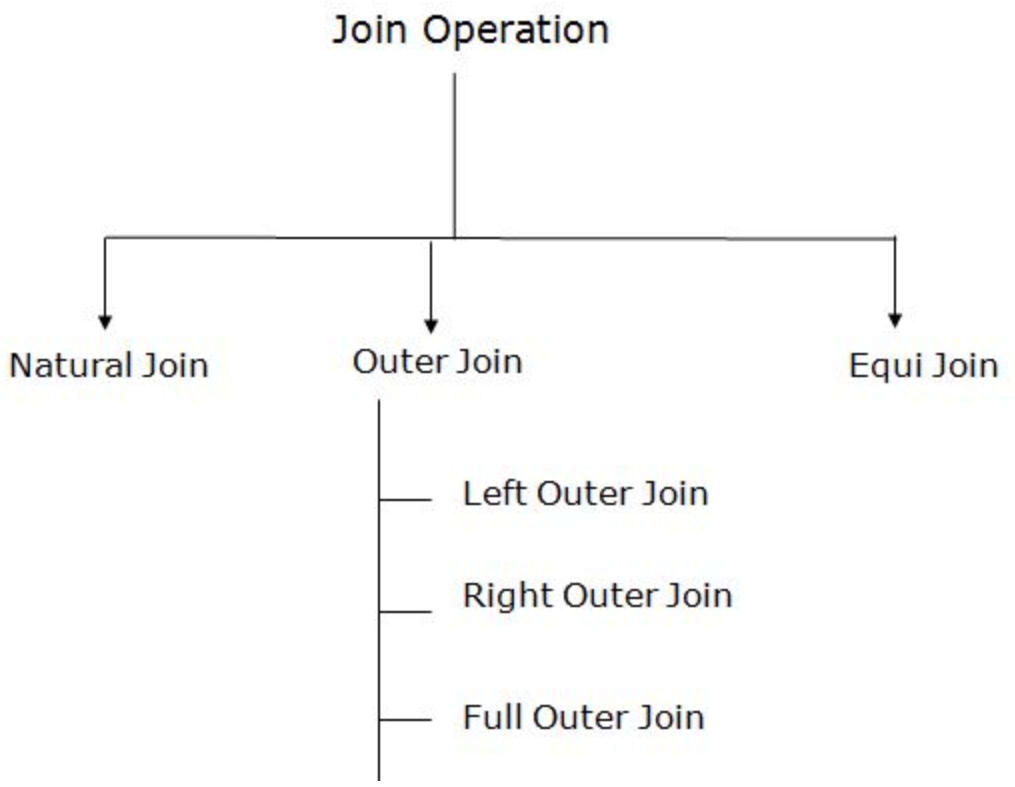
Join Operation

- Operation: (EMPLOYEE ⋈ SALARY)
- Output:



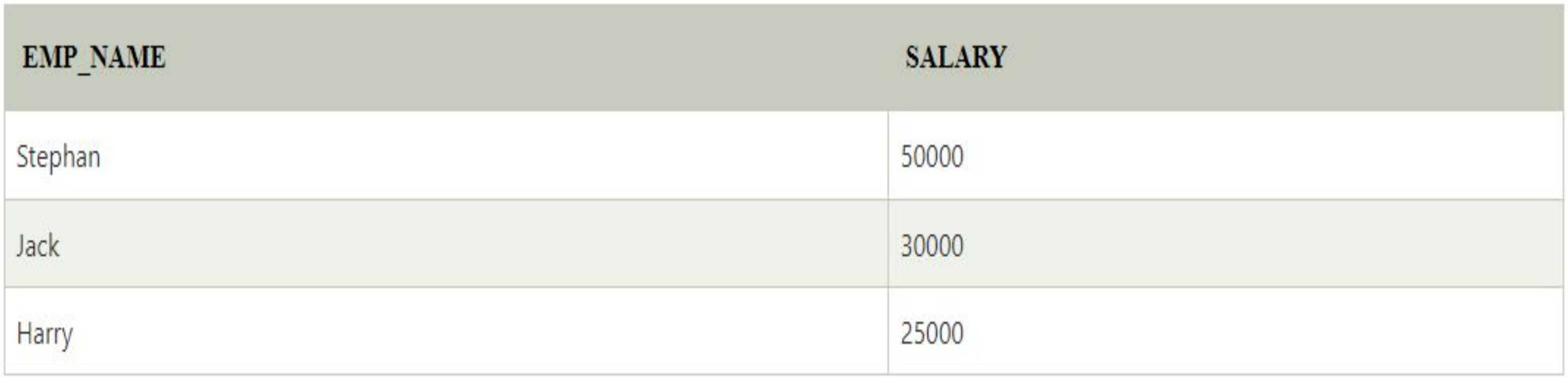
EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Types of Join operations



Natural Join

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.
- ∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)
- Output:



EMP_NAME	SALARY
Stephan	50000
Jack	30000
Harry	25000

Outer join

- The outer join operation is an extension of the join operation.
- It is used to deal with missing information.
- An outer join is basically of three types:
  - Left outer join
  - Right outer join
  - Full outer join

Outer join

- Employee Table :

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

- FACT_WORKERS :

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Outer join

- Input :
  - (EMPLOYEE ⋈ FACT_WORKERS)
- Output :

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

Left outer join

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.
- Input :
  - EMPLOYEE ⟕ FACT_WORKERS
- Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

Right outer join

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.
- Input :
  - EMPLOYEE ⟖ FACT_WORKERS
- Output :

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

Full outer join

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.
- Input :
  - EMPLOYEE ⟗ FACT_WORKERS
- Output :

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

Equi join

- It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).
- $A \bowtie_{A.column = B.column} (B)$

Example

- Consider the Following Relational schema:
  - Actor(actor_ID, a_name, nationality, age)
  - Film(film_ID, title, year, director_ID)
  - Performance(actor_ID, film_ID, character)
  - Director(director_ID, d_name, nationality)
- Construct relational algebra queries for the following statements:
  - Retrieve the names of all British directors.
  - Find out the names of all American actors above the age of 40.
  - Retrieve the name of each actor together with the titles of the films he/she has performed in.
  - Retrieve details of all films that were released in 2017.
  - Find out the names of all the actors that have played the character of Tom Cruise.

Relational Calculus

- Relational calculus is a non-procedural query language.
- In the non-procedural query language(Relational Algebra), the user is concerned with the details of how to obtain the results.
- The relational calculus tells what to do but never explains how to do.
- It uses mathematical predicate calculus.
- **Many of the calculus expressions involve the use of Quantifiers. There are two types of quantifiers:**
  - **Universal Quantifiers:** The universal quantifier denoted by ∀ is read as for all which means that in a given set of tuples, exactly all tuples satisfy a given condition.
  - **Existential Quantifiers:** The existential quantifier denoted by ∃ is read as for all which means that in a given set of tuples, there is at least one occurrences whose value satisfy a given condition.

Tuple Relational Calculus

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form

  $\{t \mid P(t)\}$

- It is the set of all tuples *t* such that predicate *P* is true for *t*
- *t* is a *tuple variable*, *t* [*A*] denotes the value of tuple *t* on attribute *A*
- $t \in r$ denotes that tuple *t* is in relation *r*
- *P* is a *formula* similar to that of the predicate calculus
- TRC is a declarative language, meaning that it specifies what data is required from the database, rather than how to retrieve it. TRC queries are expressed as logical formulas that describe the desired tuples.

Predicate Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x if true, then y is true

   $$x \Rightarrow y \equiv \neg x \vee y$$

5. Set of quantifiers:
   - $\exists\ t \in r\ (Q\ (t\ ))$ ≡ ”there exists” a tuple in *t* in relation *r* such that predicate *Q* (*t*) is true
   - $\forall t \in r\ (Q\ (t\ ))$ ≡ *Q* is true “for all” tuples *t* in relation *r*

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 6.1 The *instructor* relation.

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Electrical Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000
null	Painter	*null*

department

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

Figure 2.6 The *section* relation.

Example Queries

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000

$\{t \mid t \in instructor \land t[salary] > 80000\}$

Notice that a relation on schema (*ID, name, dept_name, salary*) is implicitly defined by the query

- As in the previous query, but output only the *ID* attribute value

$\{t \mid t \in \text{section} \land \exists\, s \in \text{instructor}\ (t[ID] = s[ID] \land s[salary] > 80000)\}$

Notice that a relation on schema (*ID*) is implicitly defined by the query

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\begin{aligned}
\{t \mid t \in \text{ section} \wedge \exists s \in \textit{section}\ (&t\,[\textit{course\_id}\,] = s\,[\textit{course\_id}\,] \wedge \\
&s\,[\textit{semester}] = \text{“Fall”} \wedge s\,[\text{year}] = 2009 \\
\vee\ \exists u \in \textit{section}\ (&t\,[\textit{course\_id}\,] = u\,[\textit{course\_id}\,] \wedge \\
&u\,[\textit{semester}] = \text{“Spring”} \wedge u\,[\text{year}] = 2010\,)\}
\end{aligned}$$

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{t \mid t \in \text{section} \land \exists s \in \textit{section}\ (t[\textit{course\_id}] = s[\textit{course\_id}] \land s[\textit{semester}] = \text{“Fall”} \land s[\text{year}] = 2009 \land \exists u \in \textit{section}\ (t[\textit{course\_id}] = u[\textit{course\_id}] \land u[\textit{semester}] = \text{“Spring”} \land u[\text{year}] = 2010)\}$$

- Find the set of all courses taught in the Fall 2009 semester, but not in the Spring 2010 semester

$$\{t \mid t \in \text{section} \land \exists s \in \textit{section}\ (t[\textit{course\_id}] = s[\textit{course\_id}] \land s[\textit{semester}] = \text{“Fall”} \land s[\text{year}] = 2009 \land \neg \exists u \in \textit{section}\ (t[\textit{course\_id}] = u[\textit{course\_id}] \land u[\textit{semester}] = \text{“Spring”} \land u[\text{year}] = 2010)\}$$

Domain Relational Calculus

Domain Relational Calculus

- A nonprocedural query language equivalent in power to the tuple relational calculus
- Each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P (x_1, x_2, \ldots, x_n)\}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables
  - P represents a formula similar to that of the predicate calculus

Example Queries

- Find the *ID, name, dept_name, salary* for instructors whose salary is greater than $80,000
  - $\{< i, n, d, s> \mid < i, n, d, s> \in instructor \land s > 80000\}$
- As in the previous query, but output only the *ID* attribute value
  - $\{< i> \mid < i, n, d, s> \in instructor \land s > 80000\}$
- Find the names of all instructors whose department is in the Watson building

$$\{< n > \mid \exists\ i, d, s\ (< i, n, d, s > \in instructor \\ \land\ \exists\ \text{b, a}\ (< d, b, a> \in department\ \land\ b = \text{“Watson”}\ ))\}$$

Example Queries

- Find the set of all courses taught in the Fall 2009 semester, or in the Spring 2010 semester, or both

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in section\ \land\ s = \text{“Fall”} \land y = 2009\ )$$
$$\lor\ \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in section\ ]\ \land\ s = \text{“Spring”} \land y = 2010)\}$$

This case can also be written as

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in section\ \land\ ((s = \text{“Fall”} \land y = 2009\ )\ \lor (s = \text{“Spring”} \land y = 2010))\}$$

- Find the set of all courses taught in the Fall 2009 semester, and in the Spring 2010 semester

$$\{<c> \mid \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in section\ \land\ s = \text{“Fall”} \land y = 2009\ )$$
$$\land\ \exists\, a, s, y, b, r, t\ (<c, a, s, y, b, r, t> \in section\ ]\ \land\ s = \text{“Spring”} \land y = 2010)\}$$